École Supérieure d'Ingénieurs
Léonard-de-Vinci

Parcours Recherche

Rapport Final

Indicateur d'Évolution des Pratiques touristiques à partir des Réseaux sociaux

Élève :
Hao Zhang

Mentor :
Jérôme Da-Rugna
Gaël Chareyron

8 avril 2018

Table des matières

1 Présentation du Sujet

Dans le cadre d'un projet de recherche financé, l'équipe Big Data du DVRC a pour objectif de mettre en place de nouveaux algorithmes à partir de données issues de réseaux sociaux :

- des algorithmes à même d'évaluer et identifier des évolutions de pratiques touristiques dans le temps ;
- des algorithmes permettant de lier ces évolutions à un e transformation territoriale majeure avec une mise en oeuvre dans le cas de la cité du Vin de Bordeaux pour la région Nouvelle Aquitaine.

Sur des données déjà collectées et une infra IT existante, le ou les élèves du parcours recherche travailleront conjointement avec les chercheurs du DVRC afin de mettre en place de nouveaux algorithmes et indicateurs et valideront eux-ci dans le cadre de la région Nouvelle Aquitaine. Les élèves seront impliqués directement dans le travail commun avec les partenaires (Bordeaux Métropole, Laboratoires spécialistes du Tourisme).

2 Étude Bibliographique

2.1 Article 1

- Titre :
Identifying Valuable Travelers and Their Next Foreign Destination by the Application of Data Mining Techniques1 (Décembre 2006)

- Extraction de Ressources :
1.la démographie des voyageurs
2.l'information sur la prise de décision des voyageurs
3.les destinations visitées les dates de la visite incluse

- Technique concernée :
1.la construction d'un arbre de décision concernant les caractéristiques du voyageur de valeur
2.le Cross-Selling Mining utilisant Market Basket Analysis (Association Rules)
3.la construction de la stratégie de promotion

- Procédure de Méthode :
1.une questionnaire pour la collection des données
2.le traitement des données (nettoyage, sélection)
3.la construction d'un arbre de décision
4.les règles d'association

5.la stratégie de promotion

• Perspective :
Certaines agences de voyages peuvent créer leur propre base de données à explorer et leur propre système de recommandation.

• Bilan :
Cette étude, elle ressemble une atypique recherche dans le domaine d'informatique, mais elle déjà utiliser les techniques ou pensés de l'exploration des données, comme le traitement de données, la construction d'un arbre de décision et les règles d'association. Donc, je pense que cet article est sans doute un bon article d'initiation.

2.2 Article 2

• Titre : Demonstrator of a Tourist Recommendation System2 (Décembre 2013)

• Extraction de Ressources :
Un GPS de suivi avec un PND (Personal Navigation Device) dans la voiture pour recueillir les activités de 12 voitures de tourisme à louer pendant plus de 14 mois en 2008/2009.

• Technique concernée :
1.les règles d'association
2.les modèles séquentiels
3.les Modèles Q
4.le centre géographique des modèles séquentiels
5.le k-Means

• Procédure de Méthode :
1.la collectionne des données en plus d'un an
2.l'organisation de positions collectées en rangées séquentielles
3.l'association de la position d'arrêt GPS et l'endroit touristique
4.cinq types d'algorithmes d'exploration de données, les règles d'association incluses

• Perspective :
1.Les organisations et agences touristiques pourraient étudier ces algorithmes pour construire un système de recommandation de leurs propres bases de données.
2.Les entreprises de suivi GPS peuvent également trouver des idées pour

améliorer l'utilisation de leurs données collectées.

• Bilan :
Les ressources extraites de cette étude sont des GPS de voiture. Par rapport au premier article, les données sont déjà numérisées et plus approchées que ce que nous étudierons. Cet article a brièvement présenté cinq types d'algorithmes d'exploration de données incluses les règles d'association, les modèles séquentiels et le k-Means, nous offert des orientations d'études pour poursuivre notre recherche au niveau algorithmique.

2.3 Article 3

• Titre :
Exploration of geo-tagged photos through data mining approaches3 (février 2014)

• Extraction de Ressources :
Des photos géolocalisées de Flickr pour le Queensland en Australie, le deuxième plus grand État de la Grande Barrière de Corail et de la forêt pluviale du patrimoine mondial.

• Technique concernée :
1.le clustering pour déterminer les paramètres pour différents niveaux et appliquer l'algorithme DBSCAN (density-based spatial clustering of applications with noise).
2.les règles d'association.

• Procédure de Méthode :
1.le prétraitement des données : supprimer les doublons et les mettre en forme pour le Clustering
2.le Clustering PoI (Points d'Intérêt) niveau mondial contre Clustering PoI au niveau local ; le Clustering PoI catégorisé.
3.les règles d'association de PoI (Points d'Intérêt) : détecter les fortes associations positives entre les PoI en cluster

• Perspective :
1.Étant donné que les photos géoréférencées capturent les traces des voyageurs, l'exploration des modèles de voyage constituant la prochaine étape à étudier.
2.La combinaison de toutes ces informations géospatiales, temporelles et textuelles pour explorer divers types de modèles est une tâche future difficile.

• Bilan :
Les ressources extraites de cette étude sont des photos géolocalisées de Flickr même quand ce que nous allons étudier. Elle utilise le clustering appliquant l'algorithme DBSCAN (density-based spatial clustering of applications with noise) et les règles d'association. Particulièrement, elle a introduit la notion des Points d'Intérêt pour identifier les lieux les plus visités d'un endroit touristique. À mon opinion, cette notion sera utilisée dans notre sujet de recherche.

2.4 Article 4

• Titre :
Detection of tourists attraction points using Instagram profiles4 (juin 2017)

• Extraction de Ressources :

L'ensemble de données recueillies contient des photos Instagram avec géolocalisation, prises par les habitants et les résidents de Saint-Pétersbourg en 2016.

• Technique concernée :
1.la méthode d'identification basée sur les fenêtres temporelles
2.le clustering hiérarchique agglomérative utilisant la distance euclidienne au carré

• Procédure de Méthode :
1.l'analyse des données
2.la détection des résidents de la ville en utilisant la méthode des fenêtres temporelles
3.le filtrage d'emplacement
4.la détection de lieux populaires en utilisant le clustering hiérarchique agglomérative

• Perspective :
Pour améliorer ce travail dans le futur, il sera possible de prendre en compte les interactions sociales entre les utilisateurs, telles que les like et les commentaires.
La présence de commentaires ne garantit pas l'approbation sociale, mais elle montre l'intérêt des utilisateurs.
Une utilisation plus poussée de l'analyse des sentiments pour les commentaires peut permettre d'approuver l'exactitude de l'évaluation des posts.

• Bilan :
Plus ressemblant à l'article dernier, cependant il utilise le clustering hié-

rarchique agglomérative utilisant la distance euclidienne au carré pour détecter les lieux populaires. Plus particulièrement, il détecte les résidents de la ville en utilisant la méthode des fenêtres temporelles, ce point vaut la peine d'apprendre.

3 Hypothèse

1 Trouver des lieux d'intéressé de la ville Bordeux et de la région Nouvelle-Aquitaine par clustering.

2 Capable de lier chaque cluster à un certain nom de lieu puis capable d'expliquer les activités d'utilisateur.

3 Si c'est possible, faire un clustering chronologique par les attributs d'années pour savoir les évolutions touristiques.

4 Trouver les potentiels liens touristiques par les règles d'association.

4 Démarche Scientifique

4.1 Sélection de Langage

Afin d'explorer cette hypothèse, j'ai choisi Python 2.7 comme langue principale pour manipuler ma recherche. Parce que, grâce à la bibliothèque libre de Scikit-learn, une bibliothèque Python dédiée à l'apprentissage automatique, le progrès de clustering de grosses données devenu plus efficace et sélectif.

Pour connecter la base de données de notre laboratoire par l'intermédiaire de Python 2, j'ai utilisé Python DB-API comme l'interface de programmation. La façon que j'utilise le tunnel ssh dans l'environnement de Python pour lier le serveur comme suit :

```
server = SSHTunnelForwarder(
    ('horus.labs.esilv.fr', 22),
    ssh_username = "prtourisme2017",
    ssh_password = "lepr2017tourisme",
    remote_bind_address = ('127.0.0.1', 3306))

server.start()

conn = MySQLdb.connect(
    host = '127.0.0.1',
    port = server.local_bind_port,
    user = 'testpano',
    passwd = 'pano01',
    db = 'touristflow',
    charset = 'utf8')
```

4.2 Prélèvement de Données

Les données de la base de données Flickr ont les attributs de latitudes et longitudes qui offrent leur information de la coordonnée géographique. Pour restreindre les données juste pour la ville Bordeaux et la région Nouvelle-Aquitaine, j'avais dessiné deux polygones du site "BoundingBox"(Figure 01), et puis j'ai obtenu les coordonnées des sommets des deux polygones. Par exemple, pour polygone de la ville de Bordeaux, les longitudes et latitudes des sommets sont :
-0.5526961264 44.7768312153,
-0.6049662525 44.7885585534,
-0.6454325561 44.7885183865,
-0.6727439786 44.8091039215,
-0.6769775873 44.8394845346,
-0.6628592252 44.8592317938,
-0.616131028 44.889303444,
-0.5696937047 44.8881289429,
-0.5329088454 44.8785441872,
-0.5191682399 44.8855425456,
-0.5105641513 44.8843003608,
-0.5022634488 44.8324474466,
-0.5526961264 44.7768312153

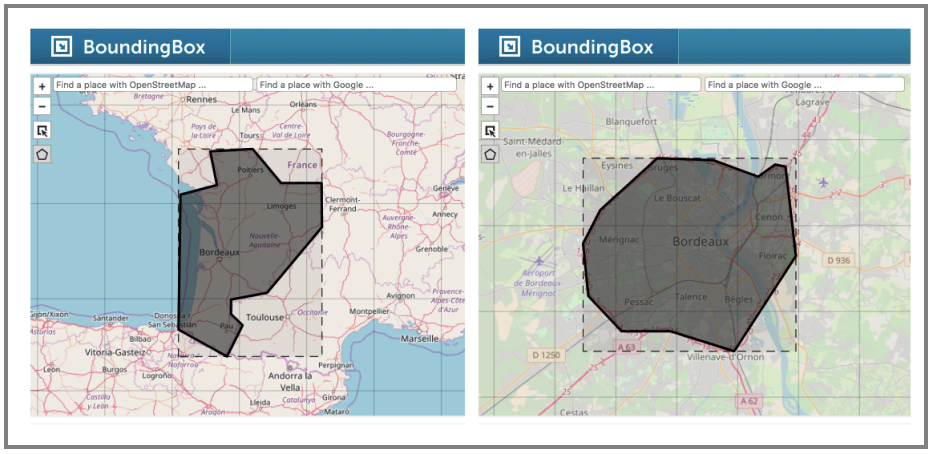

FIGURE 1 – Les polygones du site "BoundingBox"

Donc, pour la ville Bordeaux, mon commandement de SQL de prélevelment des données est :

```
1 SELECT DISTINCT owner_id, ROUND(latitude, 4), ROUND(longitude,
    4), annee, DATE_FORMAT(date_taken,'%m'), DATE_FORMAT(
    date_taken,'%d') FROM flickr_images WHERE ST_CONTAINS(
    ST_GEOMFROMTEXT('POLYGON((-0.5526961264 44.7768312153,
    -0.6049662525 44.7885585534, -0.6454325561 44.7885183865,
    -0.6727439786 44.8091039215, -0.6769775873 44.8394845346,
    -0.6628592252 44.8592317938, -0.616131028 44.889303444,
    -0.5696937047 44.8881289429, -0.5329088454 44.8785441872,
    -0.5191682399 44.8855425456, -0.5105641513 44.8843003608,
    -0.5022634488 44.8324474466, -0.5526961264 44.7768312153))'),
```

```
    POINT(longitude, latitude))
```

J'ai choisi six attributs de données qui présentent utilisateur, latitude, longitude, année, mois et jour. J'ai utilisé le commandement SELECT DISTINCT, afin que les données prélevées puissent être du même utilisateur de différentes localisations par le même jour ou être du même utilisateur de la même localisation par les différents jours. Mais, les données du même utilisateur de la même localisation par le même jour sont combinées.

Car 1 degré de latitude ou longitude est environ 111 319 km. Donc, 1 m est environ 0,00000898 degré. J'ai choisi ROUND(latitude, 4), ROUND(longitude, 4), c'est pour limiter la gamme d'erreurs entre environ 10 mètres.

40 823 données de la ville Bordeaux et 373 829 données de la région Nouvelle-Aquitaine sont prélevées pour ma recherche.

4.3 Clustering

Pourquoi faire le clustering ? C'est parce que je voudrais trouver les lieux d'intéressé. Quand les utilisateurs prennent leurs photos avec les appareils ayant la fonction de localisation, leurs informations de coordonnée géographique sont enregistrées. Mais, pour une même ville ou une même région, la densité de publication des lieux est variée. Les lieux d'intéressé sont évidemment les lieux ayant une haute densité de publication.

Donc, DBSCAN (density-based spatial clustering of applications with noise), un algorithme basé sur la densité dans la mesure qui s'appuie sur la densité estimée des clusters pour effectuer le partitionnement est un algorithme idéal pour ma recherche

Et pour améliorer la fonction, j'ai choisi à utiliser l'arbre k-d, une structure de données de partition de l'espace permettant de stocker des points, et de faire des recherches plus rapidement qu'en parcourant linéairement le tableau de points.

La Distance de Manhattan, appelée aussi taxi-distance, est la distance entre deux points parcourue par un taxi lorsqu'il se déplace dans une ville où les rues sont agencées selon un réseau ou quadrillage. Parce que l'on étudie les distances des clusters dans une ville, donc, j'ai choisi la Distance de Manhattan pour remplacer la Distance Euclidienne.

```
1 db = DBSCAN(eps = 0.0004, min_samples = 40, metric = 'manhattan
      ', algorithm = 'kd_tree').fit(X)
2
3 labels = db.labels_
4
5 n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
6
7 clusters = [X[labels == i] for i in xrange(n_clusters_)]
8
9 outliers = X[labels == -1]
```

Pour ma recherche, j'ai choisi un cluster avec un demi-diamètre d'environ 50 m (44,5 m) avec au moins 40 éléments. Et enfin, j'ai trouvé 61 clusters et 17 837 outliers pour la ville Bordeaux, 376 clusters et 295 720 outliers pour la région Nouvelle-Aquitaine(Figure 02).

4.4 Règle d'Association

Dans le domaine du data mining la recherche des règles d'association est une méthode populaire étudiée d'une manière approfondie dont le but est de découvrir des relations

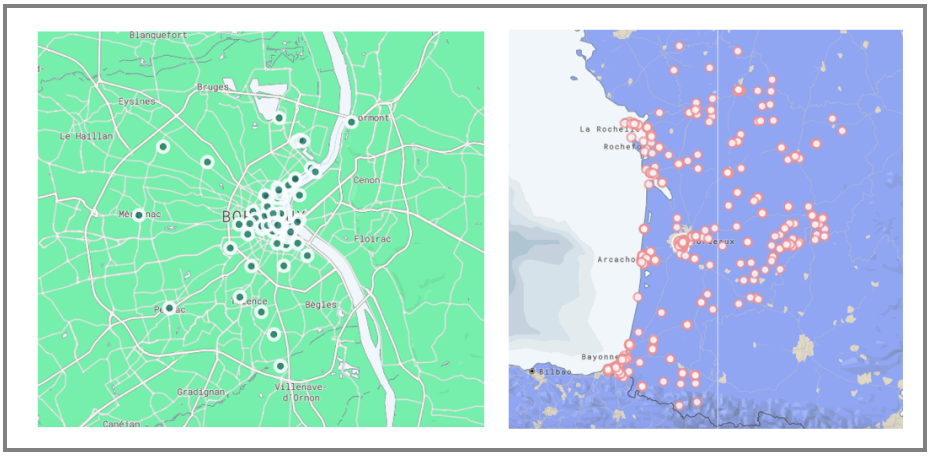

FIGURE 2 – Les clusters de la ville Bordeux (gauche) et de la région Nouvelle-Aquitaine (droite)

ayant un intérêt pour le statisticien entre deux ou plusieurs variables stockées dans de très importantes bases de données.

Pour les clusters que j'ai trouvés, chaque cluster peut avoir plusieurs utilisateurs, chaque utilisateur peut avoir plusieurs informations coordonnées corresponds aux clusters. C'est-à-dire chaque utilisateur peut visiter plusieurs lieux d'intéressé. J'ai combiné les latitudes et longitudes comme un lieu, et trouvé une liste de lieux visités pour chaque utilisateur(Figure 03).

Par l'algorithme basé d'Apriori, j'ai finalement trouvé 17 règles d'association avec au moins 0,01 de support et au moins 0,5 de conférence(Figure 04).

```
def apriori(dataset, minSupport):
    C1 = createC1(dataset)
    DataSet = map(set, dataset)
    L1, returnSupportData = scanDataSet(DataSet, C1, minSupport)
    L = [L1]
    k = 2
    while (len(L[k-2]) > 0):

        Ck = createCk(L[k-2], k)

        Lk, supportLk = scanDataSet(DataSet, Ck, minSupport)

        returnSupportData.update(supportLk)

        L.append(Lk)

        k += 1
    return L, returnSupportData

def generateRules(L, supportData, minConference):
    bigRuleList = []
    for i in range(1, len(L)):
        for subSet in L[i]:
            H1 = [frozenset([item]) for item in subSet]
```

	owner_id	latitude & longitude
0	46373861@N06	44.8366 & -0.5808
1	19614198@N00	44.8484 & -0.5722 | 44.8424 & -0.5747 | 44.836...
2	80247454@N00	44.8424 & -0.5706
3	30841592@N00	44.8424 & -0.5706
4	75363214@N00	44.8424 & -0.5706 | 44.8375 & -0.5648
5	46406392@N00	44.8424 & -0.5706 | 44.8424 & -0.5747 | 44.845...
6	81953661@N00	44.8386 & -0.5874
7	49821089@N03	44.8424 & -0.5706
8	42337233@N02	44.8484 & -0.5702 | 44.8375 & -0.5648
9	37814060@N07	44.8424 & -0.5706
10	50416032@N06	44.8379 & -0.5777 | 44.8424 & -0.5706
11	48840761@N05	44.8424 & -0.5747 | 44.8375 & -0.5648 | 44.842...
12	13435753@N03	44.8424 & -0.5706
13	23429100@N07	44.8366 & -0.5808
14	50098937@N00	44.8424 & -0.5706
15	75632071@N00	44.8415 & -0.5793
16	37998051@N00	44.8380 & -0.5752
17	49503168697@N01	44.8424 & -0.5747
18	53734320@N00	44.8500 & -0.5712 | 44.8379 & -0.5777 | 44.842...
19	11367541@N03	44.8424 & -0.5747
20	13098636@N03	44.8488 & -0.5780

FIGURE 3 – Liste de lieux visités pour chaque utilisateur

```
frozenset([u'44.8494 & -0.5790']) --> frozenset([u'44.8424 & -0.5706']) conf: 0.625
frozenset([u'44.8515 & -0.5717']) --> frozenset([u'44.8424 & -0.5706']) conf: 0.590909090909
frozenset([u'44.8311 & -0.5614']) --> frozenset([u'44.8424 & -0.5706']) conf: 0.5
frozenset([u'44.8375 & -0.5648']) --> frozenset([u'44.8424 & -0.5706']) conf: 0.66814159292
frozenset([u'44.8310 & -0.5727']) --> frozenset([u'44.8424 & -0.5706']) conf: 0.535714285714
frozenset([u'44.8484 & -0.5702']) --> frozenset([u'44.8424 & -0.5706']) conf: 0.5
frozenset([u'44.8353 & -0.5716']) --> frozenset([u'44.8424 & -0.5706']) conf: 0.511811023622
frozenset([u'44.8400 & -0.5606']) --> frozenset([u'44.8424 & -0.5706']) conf: 0.592105263158
frozenset([u'44.8488 & -0.5780']) --> frozenset([u'44.8424 & -0.5706']) conf: 0.564102564103
frozenset([u'44.8388 & -0.5741']) --> frozenset([u'44.8424 & -0.5706']) conf: 0.510638297872
frozenset([u'44.8344 & -0.5657']) --> frozenset([u'44.8424 & -0.5706']) conf: 0.558333333333
frozenset([u'44.8263 & -0.5562']) --> frozenset([u'44.8424 & -0.5706']) conf: 0.574468085106
frozenset([u'44.8424 & -0.5747']) --> frozenset([u'44.8424 & -0.5706']) conf: 0.68125
frozenset([u'44.8415 & -0.5793']) --> frozenset([u'44.8424 & -0.5706']) conf: 0.558282208589
frozenset([u'44.8424 & -0.5747', u'44.8494 & -0.5790']) --> frozenset([u'44.8424 & -0.5706', u'44.8379 & -0.5777']) c
onf: 0.705882352941
frozenset([u'44.8424 & -0.5747', u'44.8344 & -0.5657', u'44.8379 & -0.5777']) --> frozenset([u'44.8375 & -0.5648',
u'44.8424 & -0.5706']) conf: 0.565217391304
frozenset([u'44.8424 & -0.5747', u'44.8353 & -0.5716', u'44.8379 & -0.5777']) --> frozenset([u'44.8375 & -0.5648',
u'44.8424 & -0.5706']) conf: 0.583333333333
```

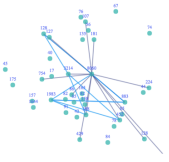

FIGURE 4 – Règles d'association avec au moins 0,01 de support et au moins 0,5 de conférence

```
25              if (i > 1):
26                  rulesFromConseq(subSet, H1, supportData,
    bigRuleList, minConference)
27              else:
28                  calculationConf(subSet, H1, supportData,
    bigRuleList,minConference)
29      return bigRuleList
30
31  def calculationConf(subSet, H, supportData,brl,minConference
    =0.05):
32      prunedH = []
33      for conseq in H:
34          conf = supportData[subSet]/supportData[subSet - conseq]
35      if conf >= minConference:
36          print subSet-conseq,'-->',conseq,'conf:',conf
37          brl.append((subSet-conseq,conseq,conf))
38          prunedH.append(conseq)
39      return prunedH
40
41  def rulesFromConseq(subSet, H, supportData, brl, minConference):
42      m = len(H[0])
43
44      if (len(subSet) > (m+1)):
45
46          Hm = createCk(H, (m+1))
47
48          Hm = calculationConf(subSet,Hm,supportData,brl,
    minConference)
49
50          if (len(Hm) > 1):
51              rulesFromConseq(subSet, Hm, supportData, brl,
    minConference)
```

5 Interprétation des Résultats

Suivi de ma recherche, j'ai pris la moyenne de latidute et longitude pour chaque cluster à remplacer les coordonnées originales des éléments. Après de ce fait, par exemple la ville Bordeaux, chacun des 61 clusters aura une seule coordonnée géographique avec elle, je pourrais trouver le lien d'un certain nom de lieu.

Avec les coordonnées des clusters, on peut facilement trouver un top 10 liste de lieux d'intéressé.

Top 10 spots of Bordeaux
Place de la Bourse - 44.8424,-0.5706
Place de la Comédie - 44.8424,-0.5747
Cathédrale Saint-André - 44.8379,-0.5777
Château du Hâ Bordeaux - 44.8366,-0.5808
Pont de Pierre - 44.8375,-0.5648
Cours de l'Intendance - 44.8415,-0.5793
Basilique Saint-Michel - 44.8344,-0.5657

Porte d'Aquitaine - 44.831,-0.5727
Église Catholique Saint-Éloi - 44.8353,-0.5716
Gare Saint-Jean - 44.8263,-0.5562

Top 10 spots of Nouvelle-Aquitaine
Bordeaux - 44.8424,-0.5706
Biarritz - 43.4834,-1.5635
La Rochelle - 46.157,-1.1525
Bordeaux - 44.8424,-0.5748
Sarlat-la-Canéda - 44.8898,1.2163
Bordeaux - 44.8379,-0.5777
Saint-Émillion - 44.8932,-0.1562
Bordeaux - 44.8366,-0.5809
Beynac-et-Cazenac - 44.8402,1.1443
Saint-Jean-de-Luz - 43.3884,-1.6639

Avec les données prélevées ayant l'attribut d'année, je pourrais faire un clustering chronologique par les attributs d'années pour savoir les évolutions touristiques, même si je ne l'ai pas encore fait.

Les potentiels liens touristiques sont déjà trouvés par les règles d'association, mais il faut faire un lien de chaque point avec un certain nom dans les étapes suivantes.